Aditya Sihag, Huzefa Raja, Jerome Thompson

Continuous Algorithms: Project Exercise

Exercise: In this exercise, you will implement the methods updateMatrix() and getMaxCorrespondingDifference() in Exercise.java.

The following conditions exist (and can be changed):

- initial temperature of the plate is 0°,
- the temperatures on the left and right is 0°,
- the temperatures on the top and bottom is 100°.

The plate is represented as a matrix. In the equation below, 'u' is the matrix.

$$u_{i,j}^{k+1} - u_{i,j}^k = r_x\left(u_{i-1,j}^k - 2u_{i,j}^k + u_{i+1,j}^k\right) + r_y\left(u_{i,j-1}^k - 2u_{i,j}^k + u_{i,j+1}^k\right)$$

Use the above equation to get the final stabilized matrix/plate. The value of 'rx' and 'ry' have already been calculated. (α = 0.000111 for copper, you can use others in the main function if you wish).

Psuedocode:

function updateMatrix()

// assume that the following global variables exist / are precalculated:

// matrix, rx, ry, and epsilon

matrixCopy = **getClone(**matrix**)** // clones (creates a deep copy of) the original matrix

for j **in** 1 to matrix.rowCount:

    for i **in** 1 to matrix.columnCount:

        matrix[j][i] = clone[j][i]

            + rx * (clone[j][i+1] - 2 * clone[j][i] + clone[j][i-1])

            + ry * (clone[j+1][i] - 2 * clone[j][i] + clone[j-1][i]);

    end for

end for

// calculate difference between cloned and new matrix values (using getMaxCorrespondingDifference)

// if difference is less than epsilon

// cancel further calculations

end function

```
function getMaxCorrespondingDifference(matrix, clone)

        // matrix is a 2D double array of dimensions m x n (usually, m=n)

        var maxError = -Double.MAX_VALUE;

    for (int i = 0; i < L; i++) { // where L is rowCount

            for (int j = 0; j < H; j++) { // and H is column count

                    double e = Math.abs(clone[i][j] - matrix[i][j]);

                    if (e > maxError && e != 0.0)

                            maxError = e;

                    end if

            end for

    end for

    return maxError;

end function
```